Deepfake Detection for Video Segments

Adam Levin
awl92@cornell.edu

Diego Montoya
dm845@cornell.edu

Gyan Suri
gs675@cornell.edu

Kieran Taylor
kct34@cornell.edu

Pablo Herrera
pah238@cornell.edu

Ryan Gale
rjg295@cornell.edu

Department of Computer Science
Cornell University
New York, NY 10044

Abstract

Deepfake techniques have made it increasingly easy to replace and manipulate a person's face within a video. The underlying technology, generative adversarial networks (GANs), are trained to produce realistic images from random seed noise. Deepfakes present an increasingly large cultural impact on trust in the legitimacy of online videos. Utilizing a large novel data set of Deepfake videos provided by AWS, Facebook and Microsoft through a Kaggle competition, we trained a binary classifier - the OffDetector - to detect facial manipulations. Using state of the art facial detection, efficient use of data batching and GPU technologies, our classifier achieves an ROC AUC of 0.973 on an unseen dataset of manipulated and original videos, the FaceForensics dataset, with an inference time of 4 seconds per video.

1 Introduction

The detection of manipulated videos can be stated as an ensemble of image and sound classification problems. Given a video (a container of synchronised images and sounds), the model should detect whether the video contains any facial manipulations. The term "Deepfake" encompasses any video altered using computer vision, deep learning or other simpler methods[2]. The Deepfake Detection challenge was set up for the public by industry, academia and civil service organizations in the technology space on Kaggle, in response to the active research and discussion ongoing at these organizations. Now that Deepfakes have matured greatly in their realism (see Figure 1), it has become increasingly difficult for humans to differentiate them from legitimate examples. The utilization of these videos ranges anywhere between benign, comedic videos of face swapped celebrities to intentional interference in the democratic process by presenting misinformation in the form of legitimate news articles[3].

The Deep Fake Detection Challenge (DFDC) data set on Kaggle consists of 119,146 high resolution (usually 1920x1080 pixels) videos of around 10 seconds each at 30 frames per second, totalling approximately 300 frames per video. In order to ensure the model generalizes well, the input data (videos) have an even distribution across race and gender, along with diversity in lighting, poses and angles of cameras. Facial attributes were inferred using computer vision techniques, and face-swaps were computed across videos to generate the Deepfakes dataset[2]. Figure 1 shows example head poses alongside their respective Deepfakes.

Our method was originally developed to compete in the DFDC Kaggle competition but the timing of the competition (which ended on March 31st) did not allow us to complete our work on time. We adapted our work to use the DFDC dataset for development and experiments and then tested

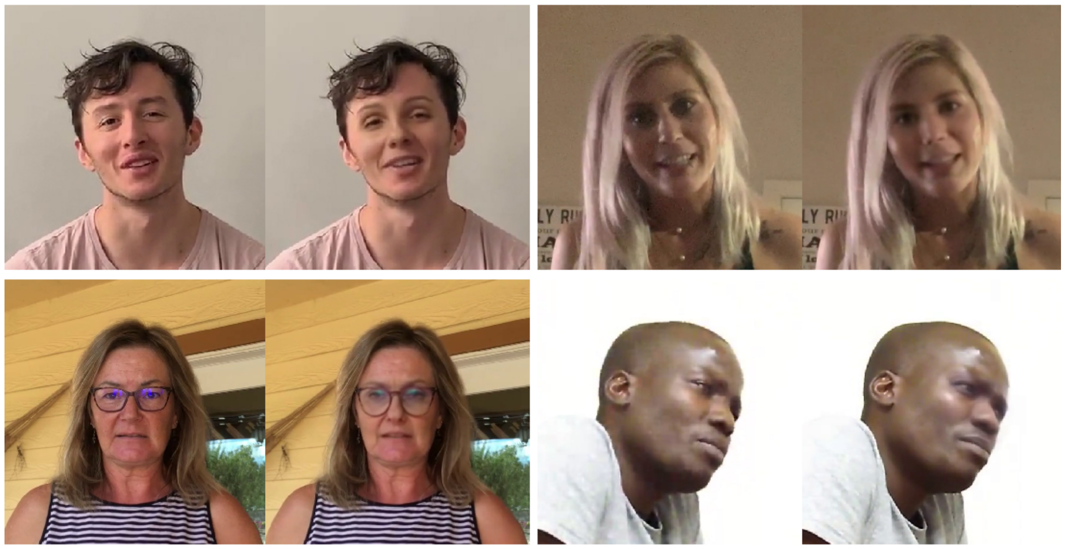
Figure 1: Original and Deepfaked Stills from Dataset comparison[2]

our system on an external, unseen dataset - the FaceForensics dataset[13]. The FaceForensics dataset was released in 2019 and contains 1000 original videos from YouTube along with 4000 manipulated videos created using four different manipulation methods. The dataset also provides three video qualities based on the compression of the videos - raw, high quality, and low quality. For our evaluation we focus on one manipulation method - DeepFakes, a method which was generated using autoencoders - and on one quality level - high quality. For this subset of the data, the FaceForesnics++ authors report an accuracy of 98.85%. However this result is obtained using a test set that was produced from the same generator model which the detector was trained on. Our method, we believe, demonstrates impressive robustness since it is trained on the DFDC dataset, which uses a variety of techniques - which are not completely revealed - for fake video generation and is distinct than FaceForensics[2], and generalizes to the FaceForensics dataset.

2 Related work

It is clear that traditional methods for detecting legitimate human faces are insufficient when faced with Deepfakes. Published just before the advent of GANs and Deepfakes, Ramachandra et al [12] reviewed the various methods for detecting 'presentation attacks', those attacks involving visual manipulations to fool a face recognition system. These focused on the most important facial recognition systems at the time, interactive identity verification such as those found at border control stations. The detection methods often exploited the interactive nature by challenging the individual to blink or move in a particular manner. These methods are unavailable in the problem of identifying Deepfakes and as Korshunov et al [7] showed, the algorithmic approaches also fall short. The equal error rate for previously very accurate systems, VGG and Facenet, rose from >1% to 88% and 95% respectively when faced with Deepfakes. However, they also showed that there are novel approaches to tackling Deepfake detection that as a baseline measure could bring the EER back below 10%.

Many of the early attempts to detect Deepfakes relied on identifying telltale artifacts produced in the generation of the videos. One particularly effective method involves the analysis of blinking patterns by the video subjects [8]. As the target face is trained predominantly on images scraped from the web and the majority of these images are open-eyed, the GAN learns to produce faces with a disproportionately low likelihood to blink. While these artifact approaches produce promising results, they present a surmountable challenge in the production of Deepfakes that would likely be solved with richer training datasets and the refinement of the training process.

Instead of focusing on pragmatic methods for identifying the weaknesses of current Deepfake generators, Agarwal and Varshney [1] aim to provide a rigorous theoretical limit on the ability of GANs to fool detection systems. They formulate the problem as the GAN learning to recreate the true distribution of facial images when

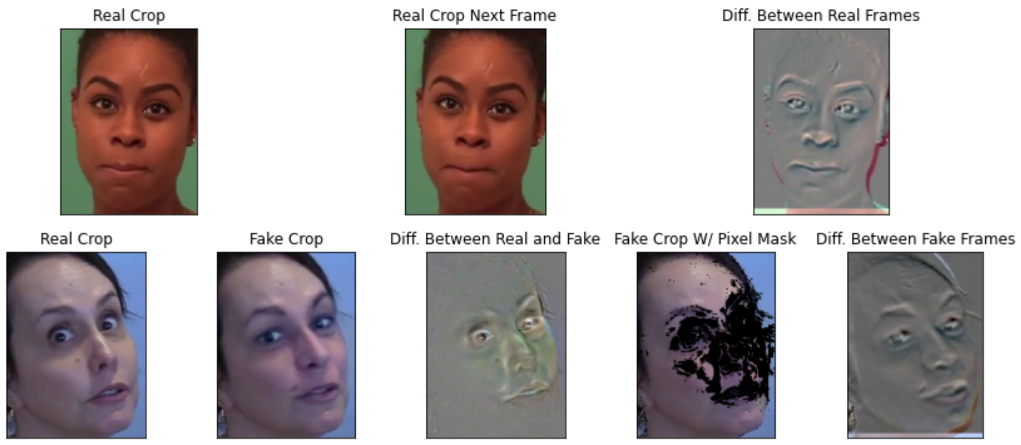
Real Crop
Real Crop Next Frame
Diff. Between Real Frames
Real Crop
Fake Crop
Diff. Between Real and Fake
Fake Crop W/ Pixel Mask
Diff. Between Fake Frames

Figure 2: A real (first row) and fake (second row) video from the DFDC dataset. Diffed images are contrast stretched for visualization purposes. The Pixel Mask is set to 0 when the difference between the real and fake image is greater than 10 (on scale of 0-255).

seeded by some random gaussian noise. In particular they note the limiting factors that increase exponentially in difficulty when training such a network. As the video resolution grows, each of the various error measures employed grows exponentially, meaning that producing high resolution Deepfakes requires an extremely accurate GAN.

Building on this generalized approach, Nguyen et al [11] recognized the need to reduce the effectiveness of easier Deepfake detection based on artifact detection. Novel approaches were used such as applying Gaussian blurs to help match the input pixel distribution more closely. They go on to identify some more inherent problems in the production of Deepfakes. The affine transformations applied to faces to match the original video result in some distortion to the resolution of the faces thus making them susceptible to CNN detectors. They break down the current Deepfake detection methods into techniques focusing on frame based artifacts and techniques combining temporal features.

Approaches using a CNN to extract frame level features that can be passed into a long short term memory (LSTM) model exposed some of the most inherently difficult parts of Deepfake generation. Sabir et al. for example, explored techniques that attempted to expose inconsistencies across frames of the videos, which led to improving the state of the art by 4.55% on the FaceForensics++ dataset [15].

On the other hand, approaches focusing on single-frame inconsistencies and artifacts are abundant and have tried attacking the problem from multiple perspectives. These approaches look for subtle details which result from the process of warping and mapping subjects in one context (resolution, lighting, pose, etc.) into another. Nguyen et al. [10], proposed a method in 2018 using capsule networks. This outperformed competing methods in several public datasets and worked for both images and videos. Other approaches work on more specific domains, like the one proposed by Matern et al. [9]. This approach looks for lighting inconsistencies in faces, especially in eyes and teeth, which arise from wrong estimation of incident light or bad estimation of geometric properties. Another such approach which works on a more specific domain is that of Koopman et al. [6], using photo response non uniformity (PRNU) — a noise pattern particular to digital camera sensors — to tell modified videos apart from real ones. They determine that PRNU analysis may be suitable for the detection of Deepfakes, however they leave the problem open as they advise that further research is needed when studying larger data sets.

3 Approach

3.1 Face Detection Pre-processing

Our training pipeline broke the problem down into two distinct steps: face detection and model training on the sequences of cropped face images. Based on a close study of the dataset which contained fake videos with their corresponding real video which was used to generate it, and with information shared in the DFDC Dataset preview paper[2], we determined that manipulations

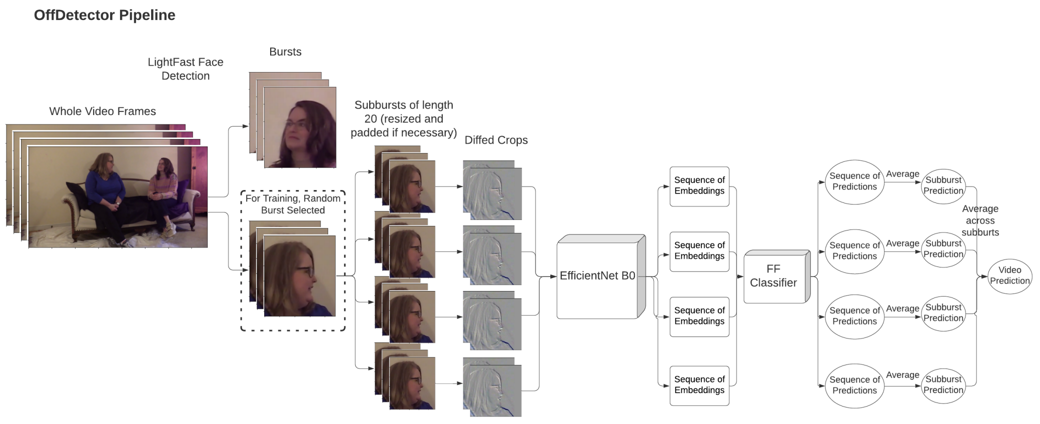

Figure 3: The OffDetector training Pipeline, end-to-end. For training, a random burst is selected to represent the video. For inference, all bursts are evaluated and the maximum predicted probability across all bursts is the final prediction.

would be focused on the faces of the actors in the videos.

With this established, we looked to extract from each frame the boxes which enclosed all faces in the frame. We explored several facial detection systems and, as this is somewhat of a solved problem, we looked to identify a tool with the best efficiency to improve our pre-processing times. LFFD[4] is optimized to run on edge devices and was able to produce our bounding boxes in reasonable time. It was also able to detect faces from profile shots and with partial occlusion.

We needed to develop heuristics for identifying individual subjects in videos with more than one face present. An upper bound had to be chosen for the intra-frame distance a bounding box could move and still be considered the same subject (i.e. if the box 'jumped' too far during a single frame we would consider it as having recognised a different face or spurious artifact). A similar choice had to be made with boxes that were dropped between frames, for example a person turning their face away from the camera for a few frames. We decided to take the simpler approach and count the broken run of frames as two separate *bursts* (sequences of the same face detected continuously). This would not affect training since during training we used short bursts, and even decomposed longer bursts into smaller ones, so the occasional broken burst was a non-issue. After these heuristics were chosen the average video in the DFDC dataset has about 1.23 bursts.

The output of the face recognition algorithm was a series of bursts of rectangle coordinates for each video, i.e. a 4-dimensional array of size *num_videos x num_bursts x num_frames_in_burst x 4*, where the last dimension contained the 4 coordinates of the bounding box of a face. This array had a set of different properties:

- *num_frames_in_burst* may change from burst to burst, being at least 15, and at most 300 (the length of a video)
- If there is a rectangle identified at frame t, the next rectangle identified at frame t' is guaranteed to have a difference $t' - t < 2$. That is, the next rectangle is guaranteed to correspond to a frame that is either the next in the video, or the next one after that. This was done in order to ensure close time correspondence between frames of the burst.
- Each burst corresponds to a single subject. i.e. there are no two rectangles in the burst that correspond to different people.

After the face detector step was finalized, the detected boxes were saved for all 119,146 videos in the DFDC dataset as text files (video timestamps along with the coordinates of faces detected). Additionally, the sequences of cropped images centered around the face were cached as PNGs. The detected boxes around faces were expanded on each side by 50 pixels (to allow margin around a face which might help the algorithm learn discontinuity artifacts) and resized to 224 by 224 pixels while preserving the aspect ratio (padding with 0 pixels as necessary). This allowed the team to experiment quickly with different model architectures as preprocessing repeatedly became computationally expensive and wasteful.

Approach	Accuracy on Val. Set
Individual Crops -> Randomly Initiated EfficientNet -> Classifier -> Averaging	63.44
Individual Crops -> EfficientNet -> Classifier -> Averaging	88.98
Individual Crops -> EfficientNet -> Two Crops -> Classifier -> Averaging	89.42
Diffed Crops -> EfficientNet -> Classifier -> Averaging	**93.68**
Individual Crops -> EfficientNet -> LSTM -> Classifier On Final Hidden State	89.31
Diffed Crops -> EfficientNet -> LSTM -> Classifier On Final Hidden State	91.90

Table 1: Experimental Results. All EfficientNet encoders are pretrained on ImageNet unless otherwise specified. Averaging refers to averaging the binary predictions to get a final prediction for the sub-burst. Two crops refers to taking successive embeddings and stacking them to double the size of the feature space. Diffed Crops refers to taking the difference of successive crops.

3.2 Model architecture

Our architecture begins by breaking the video into a variable number of bursts that are the output of the face detection pipeline. For training we select a random burst from the video and choose it to represent the video. During inference, we obtain probabilities for each of the bursts and take the maximum predicted probability as our prediction for the video. Our model, the OffDetector, works by classifying a burst, represented as N subbursts, as real or fake.

Subbursts are subsets of bursts with consecutive frames. We represent a burst as N subbursts for two reasons: it allows us to speed up training and it allows us to avoid overfitting the training data by varying which subbursts are chosen to represent a burst during each training epoch. During training time, we select two subburts to represent each burst, with randomly chosen starting indices. During inference, we select all subbursts with starting indices determined by the number of crops in the subburts and the stride of the subbursts. In our case we choose subbursts of size 20 frames, each with an overlap of 5 frames. We consider 300 frames of the input video which makes for 20 subbursts representing each burst.

For each subburst we start by taking the difference between crops in the subburst. This reduces the number of input images from 20 to 19. Taking the difference in crops empirically outperforms the same system on the crops themselves (see Experiment section). We believe the difference in crops allows the encoder to learn features that generalize more effectively by being less particular to an actor than the crops themselves. For our encoder we use EfficientNet B0. EfficientNet is a CNN that is scaled with the intent of reducing complexity and resulting number of model parameters [18]. EfficientNet is capable of matching and outperforming other state of the art architectures while maintaining a 5-10x reduction in model size. It also gives a choice of different base architectures to use for transfer learning, with each step increasing in complexity. Even the least complex B0 base architecture has been shown to perform well against common benchmarks [14].

The EfficientNet encoder returns a sequence of embeddings for each subburst. The embeddings are then classified individually as Real or Fake using a feed-forward neural network with 1 hidden layer with 1280 units. We utilize Dropout between the encoder and the classifier and between the first layer of the classifier and the hidden layer. The predicted logits are then averaged together to obtain a logit prediction for each subburst. And then finally these predictions are averaged together to obtain a predicted logit for the burst. Our entire pipeline is depicted in Figure 3.

4 Experiments

For our experiments, we standardized a subset of 7686 videos from the DFDC dataset for our training set and a different subset of 1852 as the validation set. The dataset comes in 50 different parts with most of the videos from any particular actor being in a single part. In order to make sure our models were learning in a way that would generalize across actors without learning features too specific to any one actor, we chose our validation set from a distinct set of parts (40-50) than our training set (parts 1-35). We also chose to balance our training and validation sets by undersampling the fake videos which brings the ratio for real to fake videos from about 1:5 to 1:1.

We standardize the number of subbursts to use for training (2, starting at randomly selected indices within the burst), number of subbursts for validation (13, starting at fixed, evenly spaced indices within the burst), and the number of crops in each subburst (20).

For all experiments we use the AdamW optimizer with learning rate 4e-4 and weight decay of 1e-3. We linearly reduce our learning rate by multiplying it by .7 every 3 epochs. We train until validation loss does not improve for two epochs and report the best accuracy achieved on the validation set.

Metric	Result on FaceForensics
Accuracy w/ Threshold at 0.5	80.1
Precision w/ Threshold at 0.5	62.4
Recall w/ Threshold at 0.5	97.7
F1 w/ Threshold at 0.5	76.1
Recall @ 1% FPR	79.5
Recall @ 5% FPR	89.1
Recall @ 10% FPR	92.9

Table 2: Test Results. Performance metrics of our method on the unseen FaceForensics dataset.

We experiment with two different inputs to the encoder, two different strategies for decoding the sequences of encoded embeddings into a prediction, and a few other approaches. The first input to our encoder was simply crops from the subbursts. The second is “diffed crops", visualized in Figure 2, is obtained by taking successive pairs of crops (resized to 224x224) and subtracting them. The idea was to expose our encoder to some of the temporal information contained in the video. The first decoding approach was to simply classify each of the encoded embeddings individually and then average the predictions together to get the binary prediction for each subburst. The second decoding approach was to feed the sequences of embeddings to an LSTM model (learned from random initiation) and classify the subburst using the final hidden state. We also experimented with two other approaches: stacking pairs of successive embeddings together so that the classifier can see pairs of embeddings instead of a single embedding, and feeding the subbursts of crops into a pre-trained 3D ResNet[19], modifying the fully connected layer to predict the binary Fake/Real value.

All experimental results are displayed in Table 1. The most influential contributors to the success of the model are the pre-training of the encoder on ImageNet and the Diffing of crops before feeding into the encoder. Using an LSTM as a decoder helps when feeding crops into the encoder but not when feeding in diffed crops. Perhaps the LSTM is able to recover some information over the time dimension when feeding in regular pictures which improves performance. However, when difference between frames is fed to the encoder the LSTM does not improve performance. Surprisingly (to the team), using the learned weights on ImageNet as an initialization was crucial despite when using diffed images for a completely different task.

We also experimented with the following variations, which produced inferior or not better results but were notable nonetheless:

- Instead of taking a randomly selected burst to represent the video during training, taking all the bursts from the video, getting a prediction and taking the max prediction across the bursts in a Multi-Instance Learning style. Surprisingly this never performed as well.
- Using a 3D ResNet, pre-trained for video classification, to classify subbursts as real or fake.
- Using a UNet architecture in a multitask-learning framework to predict both real/fake videos and predict pixels from fake videos with significant deviations (with a hard threshold) from the corresponding real video.
- Feeding the encoder difference of differences of crops.
- Making the difference of crops asymmetric by passing it through a ReLU non-linearity to capture only what is emerging in the next frame and not what is disappearing.
- A (we believe) novel architecture which operates on video inputs by successively doing 2D convolutions and replacing the feature maps with the difference between successive feature maps to reduce the dimension of the video in the frame dimension as it passes through the network.

5 Results

After finding the best model through experimenting on a subset of the DFDC dataset, we trained our best model on a larger subset of the data (32,456 videos) and turned the full pipeline into an API which takes a path to an MP4 file and returns a predicted probability. For computational reasons we run only the first 300 frames of the video through our pipeline, starting from the face detection and concluding with taking the maximum the predicted probabilities from our model across the bursts detected.

We ran this inference on a test set of 1000 real videos and 1000 fakes videos from the FaceForensics Dataset. Performance metrics are displayed in Table 2 and an ROC curve is displayed in Figure 4.

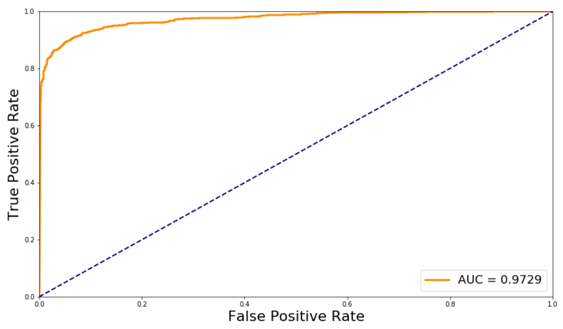

Figure 4: ROC Plot of the OffDetector on FaceForensics Dataset

Upon analyzing the results we realize that our model is not well calibrated - the average predicted probability is 0.68 even though there are half real and half fake videos in the test set, and 49.3% of the videos received a predicted probability of over 0.99. We attribute this lack of calibration to two sources: neural network classifiers are often not well calibrated to output probabilities generally, and our model learned by representing videos as a randomly selected burst but at inference we take the maximum of the predicted probabilities of being fake across all bursts which skews the output distribution. If this system were being prepared for use in a production setting, one of the steps would need to be to fit a calibration model to modulate the predicted probabilities using an external dataset. To demonstrate this need for calibration, using an optimal threshold, about 0.994, on the FF Dataset, the predictions are 92% accurate.

In terms of speed, our end to end API averages 4 second per video. Since we consider 300 frames for each video, this is 75 FPS. This test was done on a single NVIDIA V100 GPU. Speed could be improved by batching face detection model which is currently run one frame at a time. At 4 seconds per video, our system would have met the time requirement for a submission for the DFDC competition, which was about 8 seconds per video.

6 Conclusion

The final winning log-loss score of 0.42320 in the DFDC Kaggle competition [5] is a testament to how difficult this task still remains. For context, a log loss score of 0.69314 would result from simply predicting 0.5 (no preference for fake or real) on every input during inference. This is despite the DFDC data being a quality curated data set and a four month long competition. Interestingly, the public and private leaderboards differed greatly once released, which may indicate a tendency for overfitting of these approaches to the particular data set. This only highlights the need and current lack of a generalized solution to this problem.

Looking at real world applications, this could lead to an interesting approach where factually important videos such as political messaging and news broadcasts are less trusted if they are reproduced in lower quality. This would in some ways be the formalization of how we suspect many already feel about low quality online video.

One of the things that was common amongst all entrants to the competition was the lack of utilization of the accompanying audio tracks. To date none of the attempts to leverage audio information have been successful, despite the intuitively significant indicator that if the voice of the person does not match the person's face, then one of the two must have been manipulated. We think that this is due to a few reasons that may have significant implications for building a real world detection system. Most of the approaches which utilized audio information were attempting to do very low level analyses such as synchronisation of the lip movement and speech wave forms of the subject. This is an inherently difficult problem and also one which is theoretically surmountable by the refinement of generators.

If a model could be produced that was able to emphasise recall over precision, given the assumed low prior probability of Deepfakes in the wild, such a tool could be very beneficial as an automated step that triggers human intervention. Such systems already exist [17] and are widely deployed for online video platforms in flagging adult content and copyrighted materials [16]. Our results help to give an indication of what performance you could likely expect from such a flagging system and lay the foundation for a real world strategic implementation.

Finally, it was interesting to note how the addition of an LSTM didn't provide benefits to the approach. This could mean that there is still untapped potential in extracting information from temporal inconsistencies, since it's clear that this is one of the easiest features for humans to be able to tell a modified video apart from an unmodified one. Delving deeper into this approach could eventually lead to interesting improvements.

References

[1] Sakshi Agarwal and Lav R. Varshney. "Limits of Deepfake Detection: A Robust Estimation Viewpoint". In: *CoRR* abs/1905.03493 (2019). arXiv: 1905.03493. URL: http://arxiv.org/abs/1905.03493.

[2] Brian Dolhansky et al. "The Deepfake Detection Challenge (DFDC) Preview Dataset". In: *arXiv preprint arXiv:1910.08854* (2019).

[3] Luciano Floridi. "Artificial intelligence, deepfakes and a future of ectypes". In: *Philosophy & Technology* 31.3 (2018), pp. 317–321.

[4] Yonghao He et al. *LFFD: A Light and Fast Face Detector for Edge Devices*. 2019. arXiv: 1904.10633 [cs.CV].

[5] Kaggle. *Deepfake Detection Challenge*. URL: https://www.kaggle.com/c/deepfake-detection-challenge.

[6] Marissa Koopman, Andrea Macarulla Rodriguez, and Zeno Geradts. "Detection of deepfake video manipulation". In: *The 20th Irish Machine Vision and Image Processing Conference (IMVIP)*. 2018, pp. 133–136.

[7] Pavel Korshunov and Sébastien Marcel. "DeepFakes: a New Threat to Face Recognition? Assessment and Detection". In: *CoRR* abs/1812.08685 (2018). arXiv: 1812.08685. URL: http://arxiv.org/abs/1812.08685.

[8] Y. Li, M. Chang, and S. Lyu. "In Ictu Oculi: Exposing AI Created Fake Videos by Detecting Eye Blinking". In: *2018 IEEE International Workshop on Information Forensics and Security (WIFS)*. Dec. 2018, pp. 1–7. DOI: 10.1109/WIFS.2018.8630787.

[9] Falko Matern, Christian Riess, and Marc Stamminger. "Exploiting Visual Artifacts to Expose Deepfakes and Face Manipulations". In: *2019 IEEE Winter Applications of Computer Vision Workshops (WACVW)* (2019), pp. 83–92.

[10] Huy H. Nguyen, Junichi Yamagishi, and Isao Echizen. "Capsule-Forensics: Using Capsule Networks to Detect Forged Images and Videos". In: *CoRR* abs/1810.11215 (2018). arXiv: 1810.11215. URL: http://arxiv.org/abs/1810.11215.

[11] Thanh Thi Nguyen et al. *Deep Learning for Deepfakes Creation and Detection*. 2019. arXiv: 1909.11573 [cs.CV].

[12] Raghavendra Ramachandra and Christoph Busch. "Presentation Attack Detection Methods for Face Recognition Systems: A Comprehensive Survey". In: *ACM Comput. Surv.* 50.1 (Mar. 2017). ISSN: 0360-0300. DOI: 10.1145/3038924. URL: https://doi.org/10.1145/3038924.

[13] Andreas Rössler et al. *FaceForensics++: Learning to Detect Manipulated Facial Images*. 2019. arXiv: 1901.08971 [cs.CV].

[14] Rajath S, Sumukh Aithal K, and Natarajan Subramanyam. *Transfer Learning using Neural Ordinary Differential Equations*. 2020. arXiv: 2001.07342 [cs.LG].

[15] Ekraam Sabir et al. "Recurrent Convolutional Strategies for Face Manipulation Detection in Videos". In: *CoRR* abs/1905.00582 (2019). arXiv: 1905.00582. URL: http://arxiv.org/abs/1905.00582.

[16] Leron Solomon. "Fair Users or Content Abusers: The Automatic Flagging of Non-Infringing Videos by Content ID on Youtube". In: *Hofstra L. Rev.* 44 (2015), p. 237.

[17] Julian Stottinger. "Skin Paths for Contextual Flagging Adult Videos". In: *Advances in Visual Computing* (2009).

[18] Mingxing Tan and Quoc V Le. "Efficientnet: Rethinking model scaling for convolutional neural networks". In: *arXiv preprint arXiv:1905.11946* (2019).

[19] Du Tran et al. "A Closer Look at Spatiotemporal Convolutions for Action Recognition". In: *CoRR* abs/1711.11248 (2017). arXiv: 1711.11248. URL: http://arxiv.org/abs/1711.11248.